Einführung Robotik **Übung 3**

Aufgabe 1): Events und Timer ***(keine Abgabe)***

In dieser Aufgabe soll das Event- und Timer-Konzept, welches in der Vorlesung vorgestellt wurde, auf Basis von NNXT umgesetzt werden. Führen Sie hierfür bitte die folgenden Aufgaben durch:

a) Arbeiten Sie den Abschnitt „Kritische Bereiche“ im Tutorial #04 – Das Betriebssystem des NNXT durch
b) Überlegen Sie sich, wie Sie ein Event-Konzept verwirklichen: Eine Task kann ein Event (ohne Wert) erzeugen, dass von anderen Tasks konsumiert werden kann. Stellen Sie Funktionen zur Verfügung, die für die Behandlung und Erzeugung von Events benutzt werden sollen. Sie sollten mindestens die folgenden Funktionen implementieren („…“ können Sie ggf. durch weitere Parameter ergänzen):

```
void setEvent(EventType ev, …)
bool eventIsSet(EventType ev, …)
void clearEvent(EventType ev, …)
```

Nutzen Sie hierfür bitte möglichst speichersparsame Datentypen und Bitoperationen. Gehen Sie davon aus, dass maximal 16 verschiedene Events auftreten können und verwalten Sie die Events global und nicht pro Task.
c) Erstellen Sie eine Implementierung für das Konzept von allgemeinen Timern. Timer sollen dabei im Millisekundenbereich eingestellt werden können und liefern bei Ablauf der Zeitspanne ein entsprechendes Timer-Event. Die folgenden Funktionen sollten für Timer angeboten werden:

```
void setTimer(TimerType timer, int time, EventType ev)
void startTimer(TimerType timer)
void cancelTimer(TimerType timer)
```

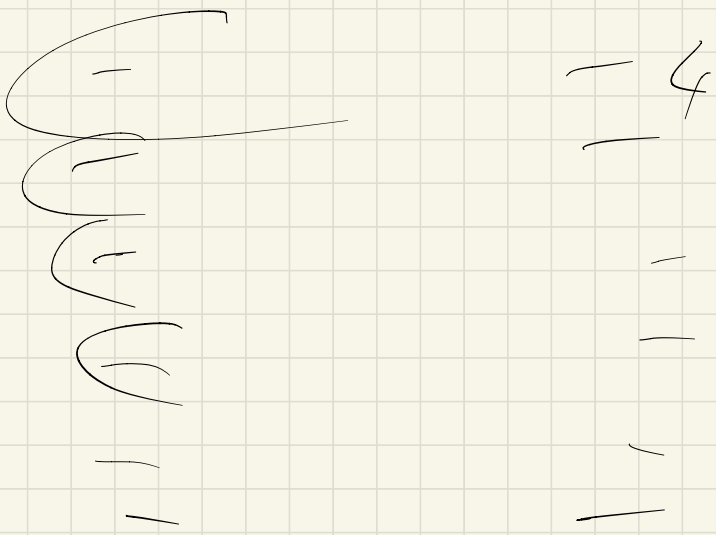
Überlegen Sie insbesondere, welche zusätzliche Infrastruktur Sie bei der Verwendung von Timern zur Verfügung stellen müssen und implementieren Sie diese.

Aufgabe 2): Events und Timer: Testbeispiel ***(Abgabe: Vorführung von Code und Anwendung)***

Als Anwendungsbeispiel nehmen Sie sich bitte die Applikation aus Übungszettel 2, Aufgabe 4) vor und bauen diese so um, dass:

a) Die beiden Tasks für die Abfrage der Taster beim Erkennen eines Tastendruckes ein Event erzeugen
b) Verändern Sie nun die Applikationstasks, so dass Sie die Events und Timer aus Aufgabe 1) dieses Übungszettels benutzen. Beachten Sie, dass Sie den Timer für die Zeitspanne von 1s irgendwo starten müssen. Benutzen Sie hierfür obige Implementierung von Timern. Achten Sie dabei darauf, dass es in NNXT nicht möglich ist, eine Task abhängig von Events zu starten, so dass Sie die Task in bestimmten Zeitabständen (z.B. 10ms) aufrufen müssen, und dann jeweils das Vorhandensein von Events testen müssen, bevor Sie tatsächlich eine Berechnung durchführen.

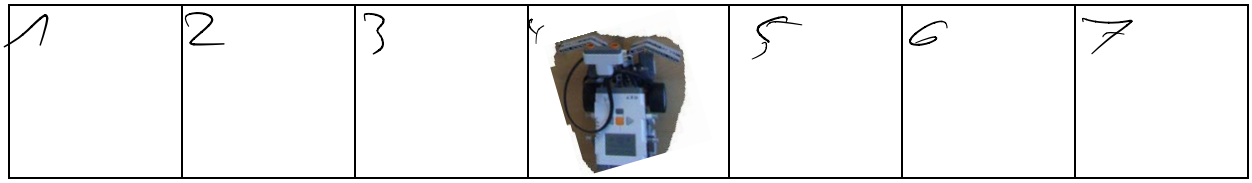
Beachten Sie auch, was passieren muss, wenn Sie während der Fahrt ein weiteres Mal den Taster drücken (Sie entscheiden das Vorgehen, bitte im Quellcode kommentieren).



100ms

Aufgabe 3): Steuerung ***(Abgabe: Vorführung von Automat, Code und Anwendung)***

In dieser Aufgabe sollen Sie eine Steuerung implementieren. Die Umwelt des Roboters besteht aus 7 gleichgroßen Quadranten (Größe können Sie wählen), die horizontal angeordnet sind. Im Startzustand befindet sich der Roboter im mittleren Quadranten mit einer vertikalen Ausrichtung. Das folgende Bild stellt dies dar:



Die Steuerung soll nun das folgende Verhalten implementieren:

- Bei einem Druck auf den rechten Taster soll der Roboter in den nächsten Quadranten rechts von der aktuellen Position fahren. Befindet sich der Roboter im letzten Quadranten, soll nichts geschehen. Die Ausrichtung des Roboters im nächsten Quadranten soll der Fahrtrichtung entsprechen.
- Bei einem Druck auf den linken Taster soll der Roboter in den nächsten Quadranten links von der aktuellen Position fahren. Befindet sich der Roboter im letzten Quadranten, soll nichts geschehen. Die Ausrichtung des Roboters im nächsten Quadranten soll der Fahrtrichtung entsprechen.
- Wird innerhalb eines Intervalls von 5 Sekunden keiner der beiden Taster gedrückt, so soll der Roboter wieder in den Startzustand zurückkehren (mittlerer Quadrant, Ausrichtung vertikal, siehe Bild).

Für die Erkennung des Tastendrucks nehmen Sie bitte die Implementierung aus Aufgabe 2 inkl. der Event-Erzeugung. Zur Implementierung bitte das folgende Vorgehen anwenden:

a) Definieren Sie die möglichen Zustände des Systems (bestehend aus Quadrant und Ausrichtung des Roboters)
b) Entwerfen Sie einen Automaten, der diese Steuerung implementiert
c) Implementieren Sie den Automaten aus b) in C. Verwenden Sie hierbei bitte die Funktionen für Timer und Events aus Aufgabe 1.

S	Taste links	Taste Rechts	Event Over
1L	1L	2R / d(180) s(1)	4U / s(-3) d(90)
2L	1L / s(1)	3R / d(180) s(-1)	4U / s(-2) d(90)
2R	1L / d(180) s(1)	3R / s(1)	4U / s(2) d(270)
3L	2L / s(1)	4R / d(180) s(1)	4U / s(-1) d(90)
3R	2L / d(180) s(1)	4R / s(1)	4U / s(1) d(270)
4L	3L / s(1)	5R / d(180) s(1)	4U / d(90)
4R	3L / d(180) s(1)	5R / s(1)	4U / d(270)
4U	3L / d(270) s(1)	5R / d(90) s(1)	4U
5L	4L / s(1)	6R / d(180) s(1)	4U / s(1) d(90)
5R	4L / d(180) s(1)	6R / s(1)	4U / s(-1) d(270)
6L	5L / s(1)	7R / d(180) s(1)	4U / s(2) d(90)
6R	5L / d(180) s(1)	7R / s(1)	4U / s(-2) d(270)
7R	6L / d(180) s(1)	7R	4U / s(-3) d(270)

Drehung im Uhrzeigersinn

s (int Segmente)
⇒ Segmente > 0 vorwärts
– " – < 0 rückwärts

d (int drehen)

drehen um drehen im Uhrzeigersinn